

C15  
131. (amended) The computer processor of claim 113, wherein the indicator elements are stored in respective entries of a table whose entries are indexed by physical page frame number.

### REMARKS

This paper responds to the Office Action of October 1, 2002 (the "October Action") and the elaboration and correction of that Office Action provided in Advisory Actions of March 2003.

Applicant respectfully requests reconsideration of the application. Claims 1-133 are now pending, a total of 133 claims. Claims 1-21 and 37-50 were allowed in the October Action. In paper no. 17, the Examiner indicated that any rejection of claims 104-112 was withdrawn. Because claim 113 recites a limitation analogous to the "necessarily disjoint" limitation of claim 104, Applicant submits that any rejection of claims 113-133 is similarly withdrawn. Of the claims not allowed, claims 22, 51, 61, 87, 94 and 96 are independent. In many cases, the rejected claims are not explicitly discussed in the October Action; instead the written rejections are incorporated by reference from the Office Action of February 20, 2002 (the "February Action").

Several of the Advisory Actions purport to relate arguments made by Applicant during telephone interviews. See, for example, paragraphs 1 and 2 of paper no. 17. These summaries of Applicant's position lack some precision, and should not be relied upon by future readers of this prosecution history as an accurate representation of Applicant's position.

#### **I. Claims 22-36, 87-93 and 96-103**

Paragraphs 9 and 31(A) of the October Action purport to reject claim 22 under § 102(b). Claim 22 is unamended, and recites as follows:

22. A method, comprising the steps of:

executing instructions fetched from first and second regions of a memory of a computer, the instructions of the first and second regions being coded for execution by computers following first and second data storage conventions, the memory regions having associated first and second indicator elements, the indicator elements each having a value indicating the data storage convention under which instructions from the associated region are to be executed;

recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a

region whose indicator element indicates the second data storage convention, and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

The Examiner's view of Richter '684 is explained for the first time in Advisory Actions mailed March 27, March 28, and April 18 (papers no. 17, 19 and 20). As explained in the accompanying Declaration of David R. Levine (Exhibit 3), ¶¶ 7-15, the Examiner's view is based on a misreading of the Richter '684 reference. The Advisory Actions attribute to Richter '684 a number of features and capabilities that are not taught in or fairly inferable from Richter '684 (Exhibit 3, ¶¶ 5, 8, 13, 14) ("These capabilities are the 'invention' of the examiner, not of Richter"), that are directly contrary to explicit teachings of Richter '684 or the knowledge in the art (Exhibit 3, ¶¶ 7, 8, 9, 12) ("Paragraph 1 [of paper no. 20] errs"), that would lead to incorrect results (Exhibit 3, ¶¶ 6, 10) (if a computer performed as assumed in the Advisory Action, "no further useful computation can be performed," and it "could not reasonably be expected to succeed"), or that would be difficult or impossible to implement (Exhibit 3, ¶¶ 6, 10, 13, 14) ("Nothing in Richter suggests that Richter even considered it possible" to operate as suggested in the Advisory Actions). Several of the capabilities assumed by the Advisory Actions are impossible even as theoretical matters; Richter '684 teaches no practical implementation of these capabilities.

When correctly understood, Richter '684 never "[alters] the data storage content ... to create a program context ... that is logically equivalent to a pre-alteration program context ..."

Independent claims 87 and 96 recite limitations similar to the "altering the data storage content of the computer to create a program context ... logically equivalent" language of claim 22, and are patentable for similar reasons.

Claims 23-36, 88-93 and 97-103 are dependent on these independent claims, and allowable therewith. These claims also recite further patentable features.

## **II. Claims 30-32, 51-86, 89-91**

The Examiner's position on claim 51 is elaborated in the Advisory Action mailed March 28, 2003 paper no. 19), ¶ 5. Claim 51 is unamended, and recites as follows:

51. A method, comprising:

storing instructions in pages of a computer memory managed by a virtual memory manager, the instruction data of the pages being coded for execution by, respectively, computers of two different architectures and/or under two different execution conventions;

in association with pages of the memory, storing corresponding indicator elements indicating the architecture or convention in which the instructions of the pages are to be executed;

executing instructions from the pages in a common processor, the processor designed, responsive to the page indicator elements, to execute instructions in the architecture or under the convention indicated by the indicator element corresponding to the instruction's page.

In paper no. 19, the Examiner quotes the definitions of "page" and of "segment" from the Rosenberg Computer Dictionary, and argues that because the some portions of the definitions are similar (not identical), the two terms can be declared "equivalent," and that the portions that are different can be ignored. There are three errors in this reasoning.

First, the Office Action purports to reject the claims under § 102(b). A § 102 rejection is only proper when the claim and the reference are identical. When there is at best "equivalence," no rejection even exists.

Second, it is impermissible to rely on a dictionary when Richter '684 has acted as his own lexicographer, and has clearly stated that "page" and "segment" are different and incompatible. col. 6, lines 62-67 (noting that segments need not be page aligned).

Third, the differences between "pages" and "segments" are well-understood in the art. These differences have been emphasized in undergraduate textbooks for nearly 30 years. Tanenbaum<sup>1</sup> writes at page 252:

[With respect to pages,] the programmer can write his programs without even being aware that virtual memory exists...

This point is crucial and will be contrasted later with segmentation, where the programmer must be aware of the existence of segments.

Those of ordinary skill in the art recognize that the difference between "pages" (as recited in claim 51) and "segments" (as discussed in Richter '684) is "crucial." They are not "equivalent,"

---

<sup>1</sup> See, e.g., Tanenbaum, Structured Computer Organization, Prentice-Hall (1976) (Exhibit 1), pages 252-253.

let alone identical.<sup>2</sup> See also Levine Affidavit (Exhibit 3), ¶ 20 (“[N]o one of ordinary skill would interpret the word ‘page’ as having any relationship to Richter’s ‘segments’”). MPEP § 2111.01 instructs that an examiner must give weight to well-established terms of art (“[T]he words of a claim must be given their plain meaning. In other words, they must be read as they would be interpreted by those of ordinary skill in the art.” emphasis added), particularly where there is no alternative available in the art. See MPEP § 2173.05(a) (where a claim is as clear as the language of the art permits, it is improper to require further clarification).

Claim 51 recites certain properties of “pages.” These properties do not exist in any structure in Richter ’684 that is even “equivalent” within the understanding in the art. Claim 51 is patentable over Richter ’684.

Claims 30, 61, and 89 recite analogous language and are patentable for analogous reasons. Claims 31, 32, 52-60, 62-86, 90 and 91 are dependent on these claims, and allowable therewith. These claims also recite further patentable features.

### III. Claim 94

Paragraphs 5 and 31(E) of the October Action discuss claim 94 in light of Richter ’684, col. 6, lines 27-39; col. 9, lines 26-57; and col. 14, lines 33-43. As now amended, claim 94 recites as follows:

94. A method, comprising the steps of:

executing a section of computer object code twice, without modification of the code section between the two executions, the code section materializing a destination address into a register and being architecturally defined to directly transfer control indirectly through the register to the destination address, the two executions materializing two different destination addresses;

the two destination code sections at the two materialized destination addresses being coded in two distinct instruction sets and, respectively, obeying the default calling conventions native to each of the two instruction sets, neither instruction set being a subset of the others.

---

<sup>2</sup> It should be noted that the term “segment” has other definitions in the art. “Pages” are crucially different than “segments” as “segment” is used in Richter ’684 and Tanenbaum. Applicant expresses no position vis-à-vis other definitions of “segment.”

Different manufacturers offer different implementations of “pages,” *e.g.*, the TLB “granularity hint” of Digital Equipment Corp’s Alpha processors. However, no such implementation known to Applicant overlaps with Richter’s use of the word “segment.”

Language analogous to the “calling convention” language of claim 94 is addressed at ¶ 26 of the October Action. There, the October Action asserts that because two different calling conventions are known in the art (even though not mentioned in Richter ’684), Richter must have implemented some mechanism for crossing between them.

This is not a permissible analysis. To make this leap, it must first be shown (a) that Richter ’684 actually uses two different calling conventions, (b) that these two different calling conventions are the calling conventions “native to each of the two instruction sets,” and (c) that Richter ’684 implemented calls from one instruction set to the other using these two calling conventions. The Office Action points to nothing in Richter ’684 in support of any of these three propositions, but instead relies on pure assumption about which Richter might have done.<sup>3</sup> Applicant traverses any reliance on assumption or Official Notice, and demands “substantial evidence” in support of this speculation, pursuant to 37 C.F.R. § 1.104(d)(2), MPEP § 2144.03, and the memorandum<sup>4</sup> from Stephen Kunin attached hereto.

From the little Richter ’684 says about analogous issues, most likely Richter implements a mode that uses the PowerPC RISC instruction set and the x86 CISC calling convention. In this mode, instructions coded in RISC instructions pass function parameters on the memory stack, not in registers.<sup>5</sup> This creates a non-default, non-native data storage convention for Richter’s RISC mode, and thus does not fall within the claim.

The existence of this alternative makes any showing of “inherency” impossible. MPEP § 2112.

Claim 94 is patentable over Richter ’684. Claim 95 is dependent thereon and adds additional patentable features.

---

<sup>3</sup> The October Action asserts that certain technologies are “traditional.” “Traditional” is not mentioned in the MPEP as an alternative test for either inherency or anticipation.

<sup>4</sup> Memorandum of February 21, 2002 from Stephen Kunin to Patent Examining Corps, “Procedures for Relying on Facts Which are Not of Record as Common Knowledge or for Taking Official Notice,” attached as Exhibit 2.

<sup>5</sup> For a description of such an embodiment, see Applicant’s specification at section IV, pages 66-67. There, Applicant discloses that some RISC code may use CISC calling conventions for parameter passing and function return values. This technique would be consistent with the other techniques disclosed in Richter ’684, and Richter ’684 says nothing to the contrary.

#### IV. A Number of Purported Rejections are Insufficient to Meet Minimum PTO Requirements

Applicant notes that the October Action deviates from MPEP requirements in several respects, and requests that these deviations not be repeated.

First, the October Action makes a number of assumptions as to what Richter '684 might have done, with no citation to Richter '684 to show what Richter '684 actually teaches.

"Traditional"<sup>6</sup> and assumption are not "substantial evidence." Rejections not supported by "substantial evidence" may not be adhered to once traversed.<sup>4</sup> In any future Office Action, Applicant requests a particular designation of particular language in prior art references in support of each limitation of any rejected claim, and in support of each assumption or inference drawn from the reference. Applicant specifically traverses the following assumptions:<sup>7</sup>

OA of 10/1/02 ¶ 3 "Richter et al. taught two calling conventions, RISC and CISC conventions."	The word "convention" appears nowhere in Richter '684.
OA of 10/1/02 ¶ 26: Richter et al. teaches transparently calling between two calling conventions.	Richter only teaches calling between two instruction set architectures; calling conventions are never mentioned
OA of 10/1/02 ¶ 26: Richter implements "traditional" features without alteration	Richter specifically teaches altering endianness of data access to a non-traditional mode

Second, no obviousness rejection in the October Action makes any showing of "reasonable expectation of success" as required by MPEP § 2143.02. For this reason, no obviousness rejections are raised. The "enablement" discussion of ¶ 11 of paper 19 is inapplicable to obviousness – in an obviousness context, it is already conceded that the reference does not disclose the invention. The law does not permit a presumption that a reference enables what it does not disclose. MPEP § 2143.02; *In re Vaeck*, 947 F.2d 488, 493, 20 USPQ2d 1438 (Fed. Cir. 1991) ("a proper analysis under § 103 requires, *inter alia*, consideration of ... whether the prior art would also have revealed that in so making or carrying out, those of ordinary skill would have a reasonable expectation of success."); MPEP § 2142 ("The examiner bears the initial burden of factually supporting any *prima facie* conclusion of obviousness," including "reasonable expectation of success").

---

<sup>6</sup> See, for example, Office Action of October 1, 2003, paragraph 26.

<sup>7</sup> These propositions are no more amenable to "inherency" than they are to Official Notice, because an alternative exists. See footnote 5 of this paper.

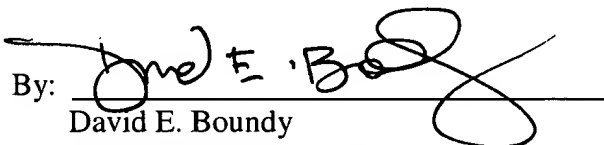
In view of the amendments and remarks, Applicant respectfully submits that the claims are in condition for allowance. Applicant requests that the application be passed to issue in due course. The Examiner is urged to telephone Applicant's undersigned counsel at the number noted below if it will advance the prosecution of this application, or with any suggestion to resolve any condition that would impede allowance. In the event that any extension of time is required, Applicant petitions for that extension of time required to make this response timely.

Kindly charge any additional fee, or credit any surplus, to Deposit Account 50-0675, Order No. 5231.03-4000.

Respectfully submitted,

SCHULTE ROTH & ZABEL, LLP

Dated: June 30, 2003

By:   
David E. Boundy  
Registration No. 36,461

Mailing Address:  
SCHULTE ROTH & ZABEL, LLP  
919 Third Avenue  
New York, New York 10022  
(212) 756-2000  
(212) 593-5955 Telecopier

- Exhibit 1      Andrew S. Tanenbaum, Structured Computer Organization, Prentice-Hall (1976), pages 252-53.
- Exhibit 2      Memorandum of February 21, 2002 from Stephen Kunin to Patent Examining Corps, "Procedures for Relying on Facts Which are Not of Record as Common Knowledge or for Taking Official Notice"
- Exhibit 3      Declaration of David R. Levine

## REWRITTEN CLAIMS MARKED UP TO SHOW CHANGES

1           1. (twice amended) A computer, comprising:  
2           a processor pipeline designed to alternately execute instructions coded for first and  
3           second different computer architectures or coded to implement first and second different  
4           processing conventions;  
5           a memory for storing instructions for execution by the processor pipeline, the  
6           memory being divided into pages for management by a virtual memory manager, a single  
7           address space of the memory having first and second pages;  
8           a memory unit designed to fetch instructions from the memory for execution by the  
9           pipeline, and to fetch stored indicator elements associated with respective memory pages of  
10          the single address space from which the instructions are to be fetched, each indicator element  
11          designed to store an indication of which of two different computer architectures and/or  
12          execution conventions under which instruction data of the associated page are to be executed  
13          by the processor pipeline, the indicator elements being architecturally addressable when the  
14          processor pipeline is executing under one of the architectures or data storage conventions,  
15          and architecturally unaddressable when the processor pipeline [computer] is executing under  
16          the other architecture or data storage convention;  
17          the memory unit and/or processor pipeline further designed to recognize an execution  
18          flow from the first page, whose associated indicator element indicates the first architecture or  
19          execution convention, to the second page, whose associated indicator element indicates the  
20          first architecture or execution convention, and in response to the recognizing, to adapt a  
21          processing mode of the processor pipeline or a storage content of the memory to effect  
22          execution of instructions in the architecture and/or under the convention indicated by the  
23          indicator element corresponding to the instruction's page.

12. (amended) The method of claim 11, wherein the indicator elements are stored in a table whose entries are indexed by [associated with corresponding] physical page frame number [frames].

27. (amended) The method of claim 22, wherein  
a rule for copying data from the first location to the second is determined by  
examining a descriptor associated with the instruction [location of execution] before the  
recognized execution flow or transfer.

31. (amended) The method of claim 30, wherein the indicator elements are stored in a  
table whose entries are indexed by [associated with corresponding] physical page frame  
number [frames].

39. (amended) The computer processor of claim 37, further comprising:  
a translation look-aside buffer (TLB); and  
TLB control circuitry designed to load the indicator elements into the TLB from a  
table stored in memory, the entries of the table being indexed by associated with  
corresponding physical page frame number [frames].

52. (amended) The method of claim 51, wherein the pages' indicator elements are  
stored in a table whose entries are indexed by [associated with corresponding] physical page  
frame number [frames], and cached in a translation look-aside buffer.

59. (amended) The method of claim 54, wherein  
a rule for copying data from the first location to the second is determined by  
examining a descriptor associated with the instruction [location of execution] before the  
recognized execution flow or transfer.

70. (amended) The microprocessor chip of claim 61, wherein the architecture or  
convention indicator elements are stored in a table whose entries are indexed by [associated  
with corresponding] physical page frame number [frames], and cached in a translation look-  
aside buffer.

90. (amended) The method of claim 89, wherein the indicator elements are stored in a table whose entries are indexed by [associated with corresponding] physical page frame number [frames].

93. (amended) The method of claim 87, wherein  
a rule for copying data from the source memory or register to the destination register or memory is determined by examining a descriptor associated with the address [location] of the control-transfer instruction.

1        94. (amended) A method, comprising the steps of:  
2        executing a section of computer object code twice, without modification of the code  
3        section between the two executions, the code section materializing a destination address into  
4        a register and being architecturally defined to directly transfer control indirectly through the  
5        register to the destination address, the two executions materializing two different destination  
6        addresses;  
7        the two destination code sections at the two materialized destination addresses being  
8        coded in two distinct instruction sets and, respectively, obeying the default calling  
9        conventions native to each of the two instruction sets, neither instruction set being a subset of  
10       the others.

97. (amended) The computer processor of claim 96, wherein the memory unit and software are designed to effect a transition between instruction boundaries, between execution in a region coded in the first instruction set using the first data storage [calling] convention to execution in a region coded in the second instruction set using the second data storage [calling] convention, so that code at the source of the flow or transfer may effect the execution transition without being specially coded for code at the destination.

98. (amended) The microprocessor chip of claim 96, wherein the two data storage conventions are first and second [two] calling conventions.

100. (amended) The microprocessor chip of claim 98 [96], further comprising:  
software and/or hardware designed to copy a datum from a first location to a second location, the first location having a use under the first calling conventions analogous to the use of the second location under the second calling conventions.

102. (amended) The computer processor of claim 98 [96], wherein  
a rule for altering the data storage content from the first calling convention to the second calling convention is determined based on an instruction at the location of execution at the source of the recognized execution flow or transfer.

103. (amended) The computer processor of claim 98 [96], wherein  
a rule for altering the data storage content from the first calling convention to the second calling convention is determined by examining a descriptor associated with the instruction [location of execution] before the recognized execution flow or transfer.

126. (amended) The computer processor of claim 118, wherein  
a rule for altering the data storage content from the first calling convention to the second is determined by examining a descriptor associated with the instruction [location of execution] before the recognized execution flow or transfer.

131. (amended) The computer processor of claim 113, wherein the indicator elements are stored in respective entries of a table whose entries are indexed by [associated with corresponding] physical page frame number [frames].